LEARNING LETTERS

Year 2026
Volume 7
Article 66
https://doi.org/10.20851/ll.v7.66

Between tool and trouble: Student attitudes toward AI in programming education

Sergio Rojas-Galeano[1], Julian Tejada[2], and Fernando Marmolejo-Ramos[3]

[1] Universidad Distrital Francisco José de Caldas, Colombia

[2] Universidade Federal de Sergipe, Brazil

[3] Flinders University, Adelaide, Australia

This study examines how AI code assistants shape novice programmers' experiences during a two-part exam in an introductory programming course. In the first part, students completed a programming task with access to AI support; in the second, they extended their solutions without AI. We collected Likert-scale and open-ended responses from 20 students to evaluate their perceptions and challenges. Findings suggest that AI tools were perceived as helpful for understanding code and increasing confidence, particularly during initial development. However, students reported difficulties transferring knowledge to unaided tasks, revealing possible overreliance and gaps in conceptual understanding. These insights highlight the need for pedagogical strategies that integrate AI meaningfully while reinforcing foundational programming skills.

Keywords: artificial intelligence, AI-based code assistants, educational technology,· programming education

Corresponding author: Fernando Marmolejo-Ramos, fernando.marmolejoramos@flinders.edu.au

Recommended citation: Rojas-Galeano, S., Tejada, J., & Marmolejo-Ramos, F. (2026). Between tool and trouble: Student attitudes toward AI in programming education. *Learning Letters, 7*, Article 66. https://doi.org/10.20851/ll.v7.66

1 Introduction

The rapid advancement of AI conversational chatbots—such as ChatGPT (OpenAI, 2023) or Perplexity—has driven transformative changes across many fields (Bubeck et al., 2023), particularly in programming and coding generation. When deployed as AI-powered code assistants these models can generate, debug, and explain code with increasing fluency. Their growing presence in professional and educational settings raises important questions about their role in computer science education.

Integrating AI coding assistants into programming curricula presents both opportunities and challenges. On the one hand, they can enhance learning by offering instant feedback, generating example solutions, and assisting with common syntactic issues (Fan et al., 2025). On the other hand, there are concerns about overreliance, reduced engagement with fundamental concepts, and weakened development of core programming skills.

Previous research in educational technology has consistently shown that the effectiveness of technological tools depends on how they are pedagogically implemented. Studies involving intelligent tutoring systems and automated feedback mechanisms indicate that these tools can enhance learning when used to complement rather than replace traditional instruction.

However, the role of AI code assistants in programming education remains insufficiently understood. While most existing research has focused on professional development contexts

or has examined AI tools in isolation, there is a growing need to explore how students perceive and engage with these tools in educational settings. Understanding these perceptions is a crucial step toward assessing the broader pedagogical implications of AI integration (Kohen-Vacs et al., 2025).

This study contributes to addressing that gap by examining student attitudes and experiences with AI assistance during a programming assignment. While not intended to measure long-term learning outcomes, the research offers preliminary insights into how students perceive the benefits and limitations of AI tools, and how such perceptions may influence their learning practices, confidence and skill development in programming.

Recent research has actively explored the impact of large language models (LLMs) on programming education for first-year college students. Alves and Cipriano (2024) analysed student interactions with GPT models during programming assignments, including object-oriented programming tasks. Becker et al. (2023) investigated how tools like ChatGPT and GitHub Copilot compare to high-performing students in CS1 (introductory computer science), highlighting implications for automated support in introductory programming. Fowler (2024) outlines some approaches to using LLMs for various assessment strategies in early programming education. Güner and Er (2025) conducted controlled experiments involving ChatGPT use, revealing different learner profiles among first-year students and the correlation of these profiles with learning outcomes. Although they conducted a well-controlled within-subjects experiment comparing student performance with and without ChatGPT support, their study was situated in a structured lab setting with instructional interventions. They found that students benefited most when using AI for code refinement rather than full generation. Our study complements this work by focusing on students' use and perceptions of AI in an authentic assessment context. By examining how novices interact with and reflect on AI assistance during real evaluative tasks, we address the gap between experimental findings and actual classroom practices.

Kulangara (2024) presented the design and evaluation of an LLM-supported educational platform aimed at teaching introductory programming, while Luo et al. (2025) assessed personalised AI mentoring using GPT systems in computing education. Shynkar (2023) explored the effects of AI-generated code on novice programmers' learning trajectories in higher education contexts.

Moon et al. (2024) examined creative coding and conversational AI interactions across experience levels, providing insight into beginner engagement with tools like ChatGPT. Although Ünlütabak et al. (2023) focused on material science education, their analysis includes perspectives on LLM applications relevant to novice programmers. Zhou et al. (2024) contributed an ethics-focused review, addressing LLM integration challenges in introductory computing courses from multiple stakeholder perspectives. This growing body of literature confirms that LLMs are reshaping the pedagogical landscape for first-year programming education, offering opportunities for both enhanced support and deeper reflection on foundational learning (Alanazi et al., 2025).

This study explores several key questions about AI-powered coding assistants in programming education. We first investigate how students' perceptions of these tools—as either "answer providers" or "learning facilitators"—affect their development of fundamental programming skills and conceptual understanding. Our research then assesses knowledge transfer, examining whether students can replicate AI-assisted work independently, and identifying the challenges they face in this transition. We also analyse engagement patterns to distinguish between deep and superficial learning, comparing outcomes for students who modify AI-generated code versus those who do not. Additionally, we document student

perspectives on the benefits and limitations of AI tools for learning, and finally, we explore their attitudes toward integrating AI into formal education, including their preferred role for AI and the desired balance with traditional instruction.

2 Methodology

2.1 Participants

The study involved 20 undergraduate students enrolled in an introductory Java-based object-oriented programming (OOP) course. While all participants had prior exposure to programming concepts, their experience with AI code assistants varied. The research was conducted in accordance with the ethical principles of the Helsinki Declaration (Wen et al., 2025), received approval from the local ethics panel, and did not collect any personal or sensitive information.

2.2 Procedure

The experiment took place during three class periods across three successive weeks in the department's computer lab (see Figure 1). Each participant attended the sessions in the following sequence:

Session 1 – Programming with AI assistance: In the first session, students were assigned to develop a complete Java application from scratch based on a medium-difficulty specification (see supplementary materials). The task involved building an object-oriented system for managing a movie theatre billboard, incorporating inheritance, dialogue-based user interfaces, and collections. Students had access to the BlueJ IDE (Kölling et al., 2003), the official Java API documentation, and were explicitly allowed to use any AI-powered code assistant of their choice (e.g., ChatGPT, Perplexity, Grok, DeepSeek). The session lasted 90 minutes and was conducted individually under supervision.

Session 2 – Programming without AI assistance: One week after the initial session, students returned for a second session focused on extending their original application. Each student received a randomly assigned programming task selected from a curated set of short, low-difficulty extensions (see supplementary materials). These tasks involved operations such as traversing object lists, applying conditional logic, and updating the state of specific objects. Implementation required a small adjustment to the existing interface—for example, by adding a new button to trigger the added functionality.

During this session, students were not allowed to use any AI assistance. They worked individually with access only to the BlueJ IDE and the official Java API documentation. To ensure compliance, all computers were offline and students were instructed to place their mobile phones face down on the desk. The session was supervised and lasted 30 minutes.

Session 3 – Post-experiment survey: In the subsequent class session, students completed an anonymous computer-based survey designed to capture their experiences and perceptions of AI use in programming tasks.

2.3 Materials

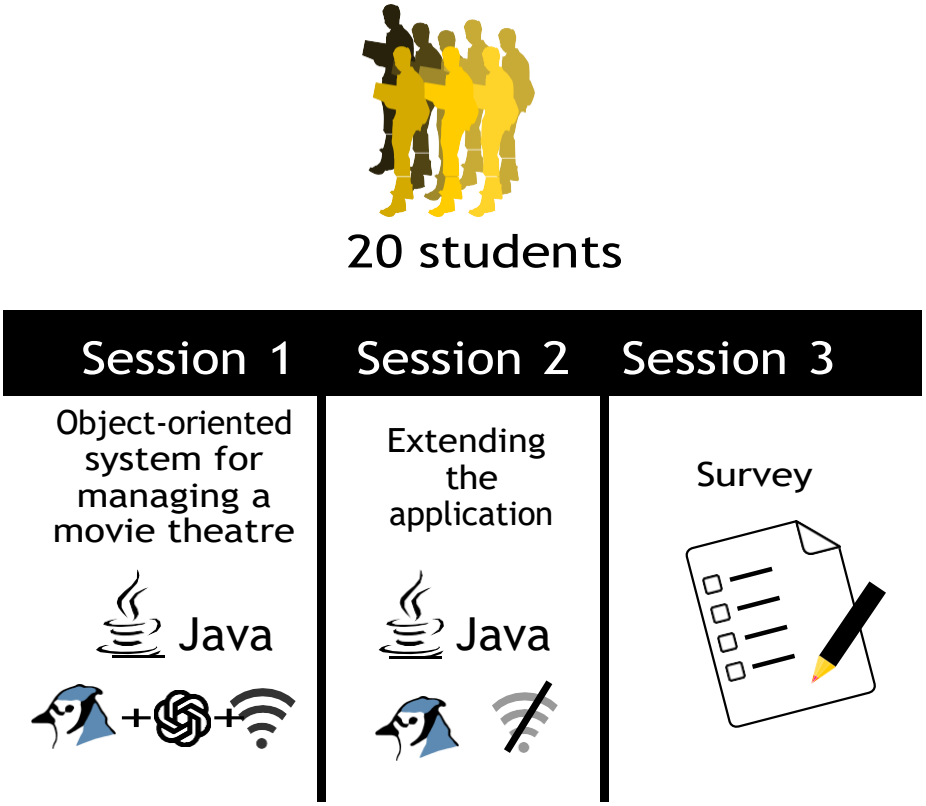
Participants underwent both main experimental conditions (coding with AI assistance and without it). During Session 3, they responded to sets of Likert scale questions and open-ended questions aimed at assessing their views and attitudes about AI, including perceived usefulness, conceptual support, and self-reported confidence levels (see supplementary materials). The survey comprised:

- Multiple choice and Likert-scale items: assessing students' programming proficiency,

AI tool usage, and perceived effects of AI on task-solving (e.g., coding speed-up, concept understanding, solution space exploration).

- Open-ended questions: exploring students' experiences in greater depth, including their understanding and adaptation of AI-generated code, perceived difficulties, general views of the exam, and how confident they felt with and without AI assistance.

Figure 1: Task development procedure overview. This scheme outlines three sessions for task development. Session 1: Developing a Java application with support from AI and the internet. Session 2: Continuing the development work without AI or internet access. Session 3: Completing a survey.



The open-ended survey questions were organised into distinct thematic sets based on their focus. Questions 8 to 10 examined students' first session experiences, specifically how they interacted with, understood, and perceived the AI-generated code. Questions 13 to 20 explored the second session, comparing AI-assisted versus non-AI performance, overall exam experiences, and exam improvement suggestions. Questions 23 and 24 addressed future-oriented perspectives, capturing students' preferences and expectations for integrating AI into programming education and learning.[1]

[1] Questions 1 through 7, 11 to 12, and 21 to 22 are of Likert type and also multiple-choice type. Although these data were analysed using techniques appropriate for categorical data analysis (Bilder & Loughin, 2017), the analyses did not yield statistically significant results. To comply with the journal's word limit, we have chosen not to include these questions in the manuscript. However, additional details can be found at https://cutt.ly/OrFdufkH and https://arxiv.org/pdf/2508.05999

2.4 Statistical analysis

Open-ended responses were examined through text mining and natural language processing (NLP) techniques via the R packages `syuzhet` for sentiment analysis and topicmodels for topic modelling using Latent Dirichlet Allocation (LDA is an unsupervised learning method that identifies topics in a collection of documents by assuming that documents are mixtures of various latent topics, Blei et al., 2003). This qualitative data was also analysed via content analysis (see supplementary files). All the R codes and datasets used in the statistical, NLP and content analyses are available as supplementary files in a companion Figshare repository: https://cutt.ly/OrFdufkH

3 Results

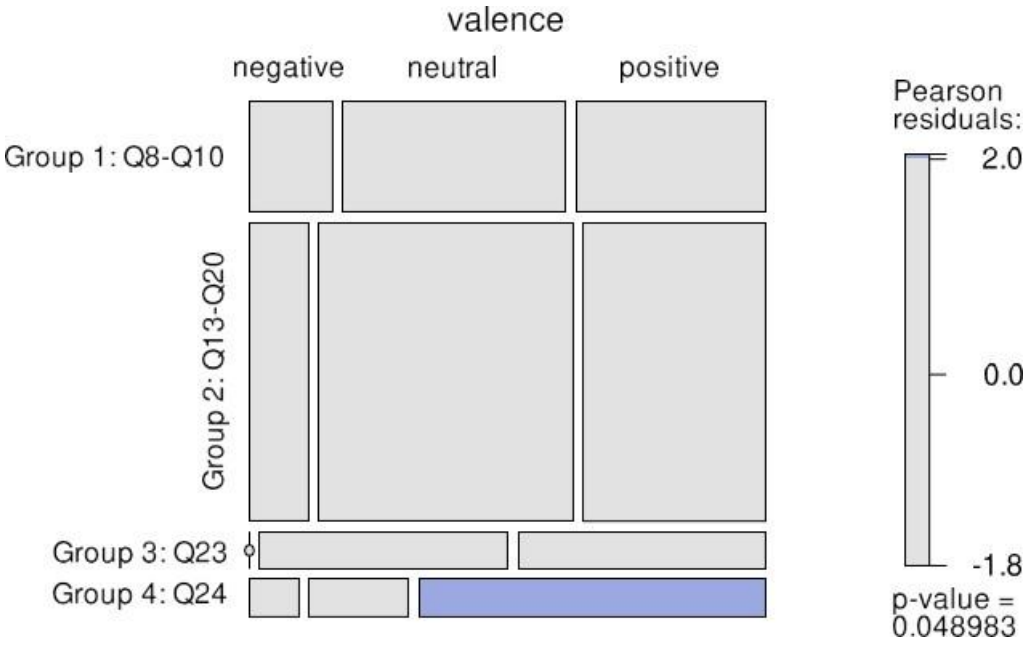
The LDA algorithm identified 2–4 latent themes within each question group, extracting the seven words with the highest β-weight (probability of appearing in that topic) for each theme. In the first group (Q8-Q10), two distinct topics emerged: troubleshooting code and conceptual clarity. This division reveals that respondents were split between addressing practical debugging issues and achieving higher-level conceptual understanding. The second question group (Q13-Q20) yielded three topics: time pressure and iteration, coding help and comprehension, and data and results. These responses demonstrated broader thematic variety, encompassing workflow timing concerns, assistance requirements, and result interpretation challenges. For Q23, two perspectives emerged: personal growth and practical application, encompassing themes such as learning gains and confidence building versus automation and project implementation. Finally, Q24 responses centered on two focal areas: future product evolution and ethical considerations, with topics including tool improvements and user expectations alongside academic integrity and originality concerns.

Content analysis using generative AI models and expert interpretation (Elo & Kyngäs, 2008; Han et al., 2025; Morgan, 2023; Nelson, 2020) revealed several key patterns. In initial AI experiences, most students successfully understood AI-generated code, particularly those with programming backgrounds. Students actively engaged by seeking clarifications, and widely modified code through refining variables, simplifying solutions, and extending outputs, though confidence levels varied regarding code efficiency and correctness. For subsequent comparisons, AI provided beneficial scaffolding through foundational understanding and reusable structures. While confidence without AI was mixed due to syntax and language barriers, students agreed AI would improve efficiency despite recognising important learning trade-offs between speed and skill development. Students found the exam format effective for assessing programming skills, appreciating the novel approach while suggesting timing and guidance improvements. Regarding future integration, students favoured a balanced approach using AI as a support tool for debugging, learning assistance, and problem-solving, while maintaining control over selective use. However, resistance emerged against AI replacing core instruction, with preferences for minimal use in formal coursework, restriction to autonomous learning contexts, or complete rejection in favour of traditional pedagogy. The key insight is that students demonstrate sophisticated understandings of AI as a supplementary tool that should enhance, not replace, fundamental learning processes (details of this content analysis are provided in the supplementary file, available at https://cutt.ly/OrFdufkH).

Finally, sentiment analysis revealed that responses to Q8-Q10 and Q13-Q20 exhibited neutral valence (median 0), with most comments containing a balanced mix of positive and negative words. Responses to Q23 (Future use of AI in programming) demonstrated a more positive tone, while responses to Q24 (AI role in programming courses) showed the highest level of

positivity overall. The analysis also identified that a small subset of responses displayed enthusiastic sentiment across all question groups, regardless of the specific topic being addressed (see details of this NLP modelling in the supplementary materials) (see Figure 2).

Figure 2: Mosaic plot illustrating the relationship between *valence of responses* (with levels positive, neutral and negative) and blocks of *open-ended questions* (Group 1:Q8- Q10, Group 2: Q13-Q20, Group 3: Q23 and Group 4: Q24).



4 Discussion and conclusion

This study explored novice programmers' perceptions of AI-powered coding assistants and their impact on foundational programming skills. Through a two-phase programming exam—first with AI assistance, then without—we assessed how these tools affect learning engagement and skill transfer. We hypothesised that students viewing AI assistants as answer providers would engage superficially and demonstrate weaker conceptual understanding, while those perceiving them as learning aids would exhibit deeper grasp of programming concepts. Our findings partially confirm this hypothesis. Students generally found AI tools beneficial for code comprehension and confidence during initial development, but struggled when applying knowledge independently, indicating overreliance and conceptual gaps. While categorical analyses yielded inconclusive results, qualitative insights revealed that students who actively modified AI-generated code reported better understanding, though this didn't consistently translate to stronger independent performance.

Students found AI particularly beneficial for syntax and structural direction initially. However, this assistance appeared mostly surface-level. Once AI tools were unavailable, many encountered difficulties applying programming concepts independently, indicating limited skill transfer. This suggests potential "cognitive offloading", where AI reliance may substitute for rather than strengthen problem-solving skills—similar to "cognitive debt", where continuous dependence on external systems could supplant challenging cognitive processes necessary

for independent thought (Kosmyna et al., 2025). The two-session design exposed critical gaps between AI-assisted performance and independent capability. While students showed initial confidence with AI support, they struggled with debugging and logic for- mulation when working alone, revealing weak internalised understanding despite strong assisted performance. Self-perception biases compounded these challenges—students underreported their AI reliance and difficulties while visibly struggling with simpler independent tasks.

These findings have broader implications for related domains like statistics education, where computational practices and conceptual reasoning are interwoven. The risk of cognitive debt is particularly salient when students rely on AI outputs without engaging in deeper analytical thinking. Wang et al. (2024) observed that while 85% of STEM students report using generative AI tools for coursework, over half tend to input problems directly and rely on generated solutions, potentially bypassing critical learning processes. To mitigate this risk, instructors should position AI as a catalyst for deeper learning rather than merely a solution provider. Effective strategies include: critically evaluating AI-generated analyses, adapting AI-produced code for different scenarios, translating AI outputs into plain-language explanations, and comparing multiple AI approaches to develop analytical judgement. The confidence and efficiency perception pathways identified by Zhang and Xu (2025) suggest that while AI tools can bolster immediate problem-solving ability, they may foster an illusion of competence. Assessment strategies should incorporate both AI-supported and independent components to help students calibrate their perceived and actual proficiency. As Wang et al. (2024) found, students who engaged with AI as a scaffold rather than a shortcut showed stronger development in problem-solving skills.

While the current results are informative, they are not without limitations. Several constraints on generality must be acknowledged (Simons et al., 2021). Specifically, sessions were presented in a fixed order (AI followed by no-AI) and were not counterbalanced. Additionally, there were differences in time-on-task (90 vs. 30 minutes) and variations in task difficulty. The sample size was also limited to 20 participants. Future research should aim to replicate and extend these findings with a larger sample and more rigorous experimental controls. Ideally, cohorts of students from courses that include programming components should be tested. Furthermore, psychometric measures should be incorporated to account for individual differences and psychological profiles.

As generative AI tools become increasingly embedded in programming education, understanding their learning impact is urgent and complex. This study provides preliminary evidence that while students value AI support and report increased confidence, these perceptions don’t always align with independent performance. Educational strategies should leverage AI for scaffolding while encouraging active engagement and critical thinking, using automation to support rather than supplant the learning process.

Acknowledgements

An extended version of this manuscript is available at https://arxiv.org/abs/2508.05999

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Conflict of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Use of AI-assisted technologies during writing

Certain paragraphs were revised with assistance from AI tools (Claude, Mistral, Gemini, or Co-Pilot) to reduce length and improve cohesion and coherence.

Availability of data and materials

All the R codes and datasets used in the statistical, NLP and content analyses are available as supplementary files in a companion Figshare repository: https://cutt.ly/OrFdufkH

Authors' contributions

S.R-G.: conceptualisation, methodology, investi- gation, supervision, project administration, writing- original draft preparation

J.T and F.M-R.: formal analysis, data curation, writing- reviewing and editing.

About the authors

Sergio Rojas-Galeano is a computer scientist and researcher specialising in artificial intelligence, machine learning, evolutionary computing, and multi-agent systems, with a recent focus on large language models and their impact on education. He is a Full Professor at the School of Engineering of the Universidad Distrital Francisco José de Caldas (Colombia), where he has led instruction in programming, algorithms, and computational intelligence for more than twenty years. He holds a PhD in computer science from University College London (UCL), awarded in 2009, and held visiting research appointments at UCL in 2022 and the University of Málaga, Spain, in 2023. A contributor to over 80 scientific publications, he is the author of two books: *Models of Learning and Optimisation for Data Scientists* (2019) and *ChatGPT: Your Python Tutor* (2023).

ORCID: https://orcid.org/0000-0002-5062-2487

Julian Tejada is a full professor of psychology at the Federal University of Sergipe, where he focuses on experimental psychology and computational modelling. His research and development interests lie at the intersection of psychology and computing. This includes developing virtual laboratories, gamifying educational activities, and creating computational models and data processing methods specifically for psychological research. An enthusiast of artificial intelligence, he actively seeks to integrate AI into his diverse courses to enhance the student experience.

ORCID: https://orcid.org/0000-0003-0275-3578

Fernando Marmolejo-Ramos is an experimental psychologist at Flinders University specialising in embodied cognition, artificial intelligence, and advanced statistical modelling. He bridges human and machine cognition through innovative research in psychometrics, AI-powered measurement, and GAMLSS-based data science. A passionate advocate for open science and interdisciplinary collaboration, his work spans from emotion processing and language embodiment to AI ethics and human–algorithm trust.

ORCID: https://orcid.org/0000-0003-4680-1287

References

Alanazi, M., Soh, B., Samra, H., & Li, A. (2025). The influence of artificial intelligence tools on learning outcomes in computer programming: A systematic review and meta-analysis. *Computers, 14(*5), 185. https://doi.org/10.3390/computers14050185

Alves, P., & Cipriano, B. (2024). "Give me the code" – Log analysis of first-year CS students' interactions with GPT. *arXiv*, Nov. https://doi.org/10.48550/arXiv.2411.17855

Becker, B. A., Craig, M., Denny, P., Keuning, H., Kiesler, N., Leinonen, J., Luxton-Reilly, A., Prather, J., & Quille, K. (2023). Generative AI in introductory programming. In *Computer science curricula 2023, Curricular practices volume*. ACM.

Bilder, C. R., & Loughin, T. M. (2017). *Analysis of categorical data with R* (2nd ed.). CRC Press.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3*, 993–1022.

Bubeck, S., Chandrasekaran, V., Eldan, R, Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Li, Y., Lundberg, S. T., Nori, H., Palangi, M., Ribeiro, M. T., & Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with GPT-4. [arXiv preprint] *arXiv*. https://doi.org/10.48550/arXiv.2506.08872

Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. *Journal of Advanced Nursing, 62*(1), 107–115.

Fan, G., Liu, D., Zhang, R., & Pan, L. (2025) The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: A comparative study with traditional pair programming and individual approaches. *International Journal of STEM Education, 12*, Article 16. https://stemeducationjournal.springeropen.com/articles/10. 1186/s40594-025-00537-3

Fowler, M. (2024). *Strategies and exercises for assessing code writing and reading ability at scale* [Doctoral dissertation, University of Illinois at Urbana-Champaign].

Güner, H. & Er, E. (2025). AI in the classroom: Exploring students' interaction with ChatGPT in programming learning. *Education and Information Technologies*.

Han, Z., Tavasi, A., Lee, J., Luzuriaga, J., Suresh, K., Oppenheim, M., Battaglia, F., & Terlecky, S. R. (2025). Can large language models be used to code text for thematic analysis? An explorative study. *International Journal of Qualitative Methods*.

Kohen-Vacs, D., Usher, M., & Jansen, M. (2025). Integrating generative AI into programming education: Student perceptions and the challenge of correcting AI errors. *International Journal of Artificial Intelligence in Education*. https://link.springer.com/article/10.1007/s40593-025-00496-4

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003) The BlueJ system and its pedagogy. *Computer Science Education, 13*(4), 249–268.

Kosmynan N., Hauptmann, E., Yuan, Y. T., Situ, J., Liao, X.-H., Beresnitzky,A. V., Braunstein, I., & Maes, P. (2025). Your brain on ChatGPT: Accumulation of cognitive debt when using an AI assistant for essay writing task. *arXiv*. https://doi.org/10.48550/arXiv.2506.08872

Kulangara, K. J. (2024). *Designing and building a platform for teaching introductory programming supported by large language models*. [Master's thesis, Aalto University, School of Science].

Luo, X., O'Connell, S., & Mithun, S. (2025). Assessing personalized AI mentoring with large language models in the computing field. In *2025 IEEE Symposium on Computational Intelligence in Natural Language Processing and Social Media*.

Moon, J. H., He, F. Q., Tian, S. S, & Hargis, J. (2024). Teaching and learning creative coding with conversational AI. *Global and Local Distance Education*,*10*(2), 3.

Morgan, D. L. (2023). Exploring the use of artificial intelligence for qualitative data analysis: The case of ChatGPT. *International Journal of Qualitative Methods, 22*, 1–10.

Nelson, L. K. (2020). Computational grounded theory: A methodological framework. *Sociological Methods & Research, 49*(1), 3–42.

OpenAI. (2023) GPT-4 Technical Report. March 2023. https://cdn.openai.com/papers/gpt-4.pdf

Shynkar, A. (2023). *Investigating the influence of code generating technologies on learning process of novice programmers in higher education computer science course* [Bachelor's thesis, University of

Twente].

Simons, D. J., Shoda, Y., & Lindsay, D. S. (2017). Constraints on Generality (COG): A proposed addition to all empirical papers. *Perspectives on Psychological Science, 12*(6), 1123–1128. https://doi.org/10.1177/1745691617708630

Ünlütabak, B., Canbek Ozdil, Z.C., & Cirit, E.S. (2023) Exploring the use of ChatGPT as a learning and teaching tool in material science and nanotechnology engineering education. *ChemRxiv*, Sept.

Wang, K. D., Wu, Z., Tufts, L., Wieman, C., Salehi, S., & Haber, N. (2024). Scaffold or crutch? Examining college students' use and views of generative AI tools for STEM education [arXiv preprint] *arXiv*. https://doi.org/10.48550/arXiv.2412.02653

Wen, B., Zhang, G., Zhan, C., Chen, C., & Yi, H. (2025). The 2024 revision of the Declaration of Helsinki: A modern ethical framework for medical research. *Postgraduate Medical Journal*, *101*(1194), 371–382. https://doi.org/10.1093/postmj/qgae181

Zhang, C., & Xu, J. (2025). The paradox of self-efficacy and technological dependence: Unraveling generative AI's impact on university students' task completion. *The Internet and Higher Education, 65*(100978). https://doi.org/10.1016/j. iheduc.2024.100978

Zhou, K. Z., Kilhoffer, Z., Sanfilippo, M., Underwood, T., Gumusel, E., Wei, M., Choudhry, A., & Xiong, J. (2024). "The teachers are confused as well": A multiple-stakeholder ethics discussion on large language models in computing education. arXiv. https://doi.org/10.48550/arXiv.2401.12453